Programación 2

Tecnicatura en Desarrollo de Aplicaciones Informáticas

Segundo Parcialito

¿Cuál es el resultado del invocar el siguiente método en java con el parámetro "Luis"

```
public String getMensajeOptimista(String nombre){
   return nombre+ " va a promocionar programación 2";
   System.out.println(nombre);
}
```

- a. Imprime por consola "Luis va a promocionar Programación 2"
- b. El código ejecuta, pero no hay salida por consola
- c. Error de compilación
- d. Error en tiempo de ejecución

¿Cuál de las siguientes frases es verdadera para una clase X?

- [x] a. Las variables, métodos y constructores que son declarados públicos en X pueden ser accedidos desde cualquier clase.
- [x] b. Las variables, métodos y constructores que son declarados privados en X pueden ser solo accedidos desde X.
- [ ] c. Las variables, métodos y constructores que son declarados privados en X pueden ser accedidos desde X y desde cualquiera de sus subclases.

Dado el siguiente código

```
public class Dado {
    private int nroCaras;

    public Dado(int nroCaras) {
        this.nroCaras = nroCaras;
    }

    public int tirar(){
        return  (int)(1+Math.random()*nroCaras);
    }

    public int getNroCaras() {
        return nroCaras;
    }

    public void setNroCaras(int nroCaras) {
        this.nroCaras = nroCaras;
    }
}
```

```
public class DadoCargado extends Dado{
    public static final int VAL_MIN = 3;

    public DadoCargado(int nroCaras) {
        super(nroCaras);
    }

    public int tirar(){
        int valor = super.tirar();
        if (valor<VAL_MIN) return valor;
        else return super.getNroCaras();
    }

    public static void main(String[] args) {
        DadoCargado dc= new DadoCargado(6);
        for (int i = 0; i < 20; i++) {
            System.out.println(dc.tirar());
        }
```

- a. No compila
- b. Compila pero da error en tiempo de ejecución
- c. Compila pero tiene un error conceptual
- d. No tiene ningún error
- e. Compila pero se cuelga en tiempo de ejecución

En base al siguiente enunciado responda:

En una empresa se desea crear un sistema para organizar a sus empleados y clientes.

De los empleados se almacena su nombre, apellido, dni, un número de empleado y el sueldo mensual. Sin embargo algunos empleados cobran ac
pc

Pc
ob
m

Lo
fre

Cliente no hereda de ninguna clase	Falso
Cliente hereda de empleado	Falso
EmpleadoConComision no debe existir como clase, es una instancia de la clase Empleado	Falso
Se debe crear la clase EmpleadoConComision	Verdadero
Empleado es una clase hija de EmpleadoConComision	Falso
Cliente hereda de Persona	Verdadero
Se debe crear una clase Persona	Verdadero
La clase Cliente no existe	Falso
EmpleadoConComision es una clase hija de Empleado	Verdadero

Dado el siguiente código, responda a las preguntas que se enuncian a continuación

```
public class X {
    public int x1(){
        return 10;
    }
    public int x2(){
        return 20;
    }
    public int x3(){
        return this.x1()+this.x2();
    }
}
```

```
public class Y extends X{
    public int x2(){
        return 200;
    }
    public int x4(){
        return 10;
    }
}
```

¿Qué imprime el siguiente código?

```
X aux = new Y();
System.out.println(aux.x4());
```

No compila

¿Qué imprime el siguiente código?

```
Y aux = new Y();
System.out.println(aux.x4());
```

10

¿Qué imprime el siguiente código?

```
Y aux = new X();
System.out.println(aux.x1());
```

No compila

¿Qué imprime el siguiente código?

```
X aux = new Y();
System.out.println(aux.x1());
```

10

Dado el siguiente código, responda a las preguntas que se enuncian a continuación

```
public class X {
    public int x1(){
        return 10;
    }
    public int x2(){
        return 20;
    }
    public int x3(){
        return this.x1()+this.x2();
    }
}
```

```
public class Y extends X{
    public int x2(){
        return 200;
    }
    public int x4(){
        return 10;
    }
}
```

¿Qué imprime el siguiente código?

```
X aux = new X();
System.out.println(aux.x3());
```

30

¿Qué imprime el siguiente código?

```
X aux = new Y();
System.out.println(aux.x3());
```

210

¿Qué imprime el siguiente código?

```
Y aux = new Y();
System.out.println(aux.x3());
```

210

¿Qué imprime el siguiente código?

```
Y aux = new X();
System.out.println(aux.x4());
```

No compila

Si la clase "Mascota" NO implementa el método "equals", el siguiente código

Ninguna opción es correcta

- Da error de compilación
- Siempre retorna false
- Imprime por pantalla "error el método equals no existe en la clase persona"
- Ninguna opción es correcta
- Siempre retorna true

```
        return false;
        return false;

}
```

Puedo invocar a super() desde cualquier línea de código del constructor de una clase

Seleccione una:

- ○ Verdadero
- ● Falso

¿Qué significa sobreescribir un método?

- ● a. Crear un método en una subclase con la misma signatura que en la clase superior
- ○ b. Editarlo en la misma clase para modificar su comportamiento
- ○ c. Cambiarle el nombre dejándolo con la misma funcionalidad
- ○ d. Crear un método con el mismo nombre pero diferentes argumentos

¿Gracias a que concepto visto (además de herencia) este código funciona para la Clase X y cualquier clase que herede de X?

```
public void imprimir(X dato){
    System.out.println(dato);
}
```

- [ ] a. A que Java tiene tipado estático
- [ ] b. static
- [ ] c. Ese código no funciona
- [x] d. Binding dinámico
- [x] e. Polimorfismo
- [ ] f. Ese código solo funciona para la clase X

Considere las siguientes dos clases, A y B. ¿Cuáles de las siguientes afirmaciones son correctas?

```
public class A {
    private int numerito;

    public A(int numerito) {
        this.numerito = numerito;
    }

    public int getNumerito() {
        return numerito;
    }

    public void setNumerito(int numerito) {
        this.numerito = numerito;
    }
}
```

```
public class B extends A {

    public B(int numerito) {
        super(numerito);
    }

    public int metodo(){
        return this.numerito*2;
    }
}
```

- a. Una invocación al método public int metodo() retorna el doble del valor asignado al atributo "numerito"
- b. Da error de compilación
- c. Da error en tiempo de ejecución

Es posible invocar super() desde cualquier método de la clase

Seleccione una:

- ○ Verdadero
- ● Falso

Cuando ejecuto la siguiente línea:

Producto leche = new Producto("Leche", "Sancor", 110);

- ○ a. Ninguna opción es correcta
- ○ b. Estoy haciendo binding dinámico entre la variable llamada leche y una clase Producto
- ● c. Estoy instanciando un objeto de la clase Producto, referenciado por una variable llamada leche
- ○ d. Estoy indicando que la Leche es un Producto mediante el mecanismo de herencia

En base al siguiente código responda:

```
public class Persona {

    private String nombre;
    private String apellido;
    private String dni;

    public Persona(String nombre, String apellido, String dni) {
        this.nombre = nombre;
        this.apellido = apellido;
        this.dni = dni;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getApellido() {
        return apellido;
    }

    public void setApellido(String apellido) {
        this.apellido = apellido;
    }

    public String getDni() {
        return dni;
    }
}
```

```
public class Empleado extends Persona{
    private int numero;
    private double sueldo;
    public Empleado(String nombre, String apellido, String dni, int numero, double sueldo) {
        super(nombre, apellido, dni);
        this.numero = numero;
        this.sueldo = sueldo;
    }

    public int getNumero() {
        return numero;
    }
    public void setNumero(int numero) {
        this.numero = numero;
    }
    public double getSueldo() {
        return sueldo;
    }
    public void setSueldo(double sueldo) {
        this.sueldo = sueldo;
    }

}
```

```java
public class EmpleadoComision extends Empleado{
    private int cantVentas;

    private double comisionPorVenta;


    public EmpleadoComision(String nombre, String apellido, String dni, int numero, double sueldo, double comision) {
        super(nombre, apellido, dni, numero, sueldo);
        this.comisionPorVenta = comision;
        this.cantVentas = 0;
    }

    public int getCantVentas() {
        return cantVentas;
    }
    public void addVenta() {
        this.addVentas(1);
    }

    public void addVentas(int cantVentas) {
        this.cantVentas += cantVentas;
    }

    public double getSueldo() {
        //suponer implementación correcta (pregunta 4)
    }


}
```

```java
public class Cliente extends Persona{

    private double compras;
    public Cliente(String nombre, String apellido, String dni) {
        super(nombre, apellido, dni);
    }
    public double getCompras() {
        return compras;
    }
    public void addCompra(double compra) {
        this.compras += compra;
    }

    public boolean recibeDescuento() {
        return this.getCompras() >= 5000;
    }
}
```

1) Se puede ejecutar el siguiente main:

```
public static void main (String[] args) {
    Persona persona = new Persona("José", "Hernandez", "12345678");
    persona.getSueldo();
}
```

○ Si

● No

2) En el caso del empleado con comisión, si el valor de la comisión puede ser diferente para cada empleado

Está bien que sea un atributo

3) ¿Cómo implementaría el método getSueldo() en la Clase EmpleadoConComision?

return super.getSueldo() + (this.getComisionPorVenta()*this.getCantVentas());
return this.getCantVentas()*0,1;
return this.getComisionPorVenta()*this.getCantVentas();
return super.getSueldo()*(this.getComisionPorVenta()*this.getCantVentas());
return super.getSueldo() + super.getSueldo()*(this.getCantVentas()*0,1);

Si una clase no hereda explícitamente de ninguna otra, invocar super() en la primer linea de su constructor da error de compilación

Seleccione una:

- ○ Verdadero
- ● Falso

¿Cuál de los siguientes es el valor por defecto de una variable local en JAVA (si no se inicializa)?

- ○ a. 0
- ○ b. No se les asigna un valor por defecto
- ○ c. null
- ● d. Depende del tipo de variable

Dadas las siguientes clases

```
public class Persona {
   private String nombre;
   public Persona(String nombre) {
      this.nombre = nombre;
   }
   public String toString() {
      return "Nombre: "+nombre;
   }
}
public class Empleado extends Persona{
   private int legajo;
   public Empleado(String nombre, int legajo) {
      super(nombre);
      this.legajo = legajo;
   }
   public String toString(String empresa) {
      return super.toString() + " legajo: "+
            legajo + " empresa: "+empresa;
   }
}
```

Cuál es el resultado de la ejecución del siguiente método ***main***

```
public static void main(String[] args) {
      Empleado empl = new Empleado("Carmelo Garcia", 1234);
      System.out.println(empl);
}
```

- a. Alumno@12cf34523
- b. Nombre: Carmelo Garcia legajo: 23457 empresa:
- c. Nombre: Carmelo Garcia
- d. Nombre: Carmelo Garcia legajo: 23457 empresa: null

Suponiendo que la clase Docente hereda de la clase Personal, ¿el compilador de Java permite que a una variable declarada del tipo Personal le asigne una instancia de la clase Docente?

- a. Depende
- b. No, nunca
- c. Si, siempre

DOCENTE

Suponiendo que la clase Docente hereda de la clase Personal, ¿el compilador de Java permite que a una variable declarada del tipo Personal le asigne una instancia de la clase Docente?

Persona

- a. No, nunca
- b. Si, siempre
- c. Depende

Clase Abstracta

Clase Abstracta : Abstracción

Normalmente usamos las abstracciones para poder referirnos a algo enfocándonos en el aspecto que nos interesa y no en detalles innecesarios

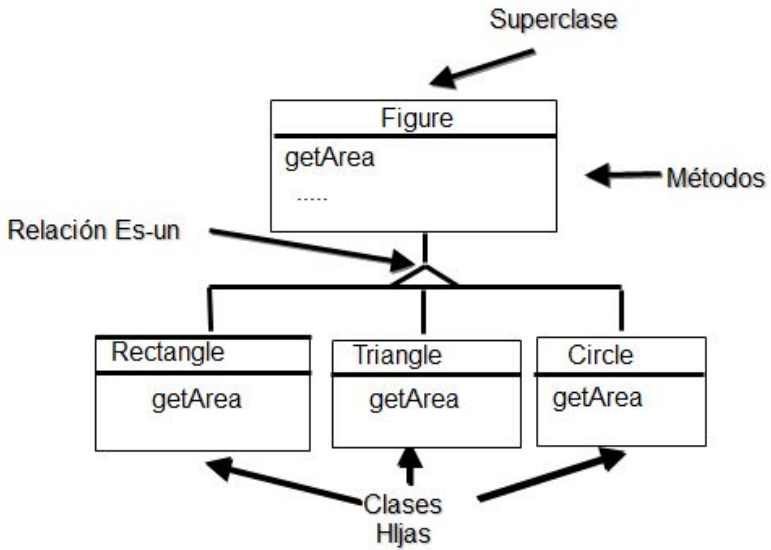
Ejemplo: el concepto de mueble, si digo “para mudarme tengo que llevar todos los muebles”, me enfoco en lo que quiero decir y no en el listado de cada mueble y sus detalles (tipo de mueble, forma, color, etc)

Clase Abstracta

En POO se pueden definir clases que representan un concepto abstracto

El concepto de Figura geométrica
es una clase Abstracta

Los conceptos abstractos no pueden
 ser instanciados



Clase Abstracta

Puedo tener instancias de una Figura?

La respuesta es NO, porque como se calcula su área? cual es su perímetro? El concepto de figura me abstrae comportamiento común y características que deben tener las figuras.

El círculo es un elemento concreto al cual le puedo preguntar su area, su perímetro

Clase Abstracta

Una clase abstracta es una clase puede ser extendida, pero no se pueden crear instancias **(no se le puede hacer new)**

Se usa la palabra clave **abstract** en la declaración

```
public abstract class Figura {  . . . . }
```

Clase Abstracta

NO se puede crear una instancia de una clase abstracta, si se lo intenta se genera un error de compilación

```
Figura   ff = new Figura();
```

Clase Abstracta : Método abstracto

Una clase abstracta además de métodos (concretos) y atributos posee **métodos abstractos**

Un Método abstracto define comportamiento común de todos los objetos de las subclases concretas de la clase abstracta

```
public abstract class Figura {

  public abstract double getArea();

..
```

Clase Abstracta: Método Abstracto

Si la clase posee un método abstracto, **la clase es una clase abstracta** y por ende debe declararse como abstracta

Clase Abstracta: hijos

Una clase que hereda de una clase Abstracta debe implementar **TODOS** los métodos de la clase de la que hereda o debe declararse abstracta

```
  public class Circulo extends  Figura {

      // Circulo Debe implementar todos los métodos abstractos de figura, getArea y getPerimetro
```

Clase Abstracta: hijos

```
public abstract class FigAreaFija extends Figura {

    double areaFija;

    public double getArea(){   //Implementa getArea que era abstracto
        return areaFija;
    }
   //Como no implementa getPerimetro, que tambien era abstracto debe declarse
abstracta la clase

}
```

Clase Abstracta: hijos

```
public class FiguraFija extends FigAreaFija {

//Solo debe implementar un método abstracto getPerimetro, el otro ya lo implemento FigAreaFija

    double perimetro;

    public double getPerimetro() {

      return perimetro;

    }

}
```

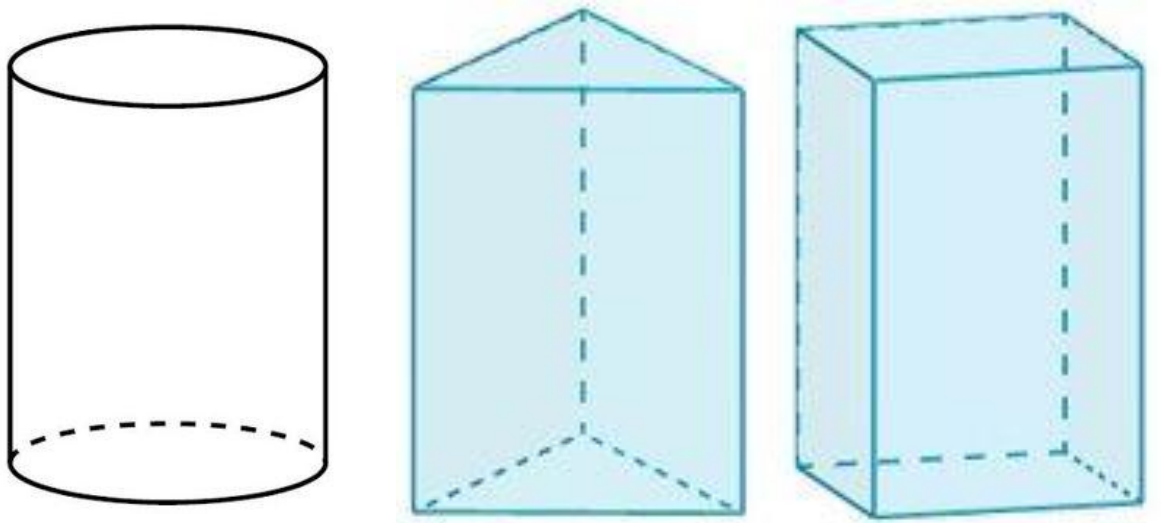
Clases Abstractas: Cuidado

Conceptualmente en la Programación Orientada a Objetos **NO** deben existir clases abstractas que no tengan al menos un método abstracto.

Esto se aclara porque Java permite clases abstractas sin métodos abstractos (también deja poner atributos publicos……)

Ejemplo Figuras3D

Figura3D



¿Qué cambia de la implementación realizada?

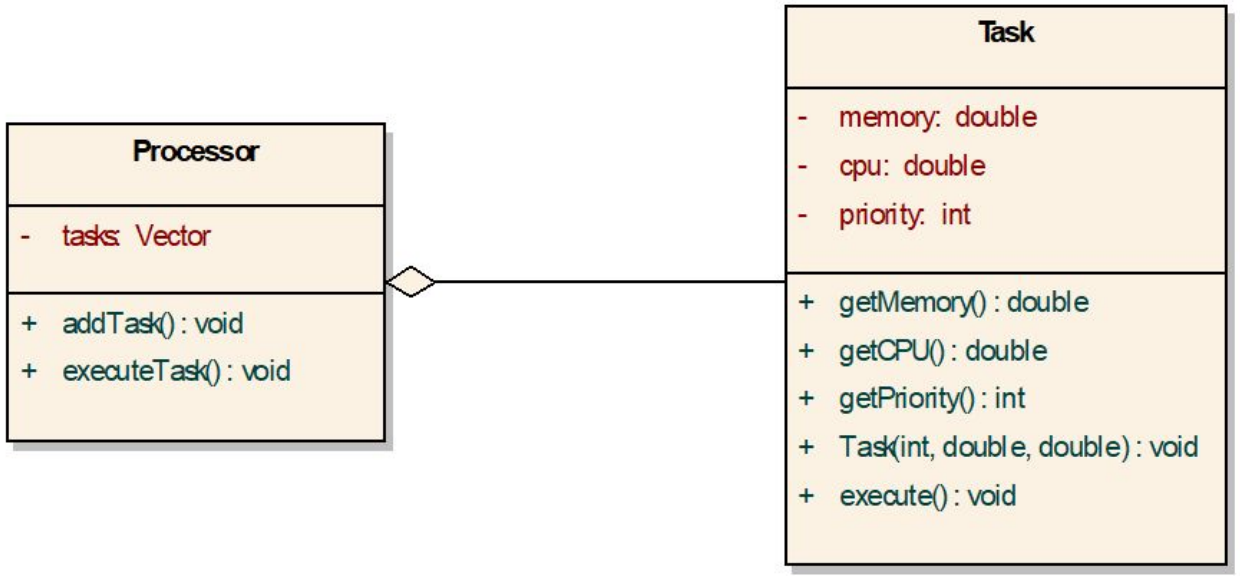
Ejemplo de Procesadores

Ejercicio

A un procesador se le asignan tareas, las mismas poseen una prioridad, uso de memoria y uso de CPU. El procesador X ordena las tareas de acuerdo a la prioridad de las mismas, mientras que otros procesadores las ordenan por uso de CPU o uso de memoria.

Procesadores

Primer aproximación a la solución



Procesadores : Clase Processor

```
public class Processor {
  Vector tasks;
  public Processor() {
     tasks = new Vector();
  }

  public void execute() {
    if (tasks.size()>0) {
      Task next = (Task)tasks.elementAt(0);
      tasks.removeElementAt(0);
      next.execute();
    }
```

Procesadores : Clase Processor 2 Parte

```
 ---
public void addTask(Task task) {
    int i = 0;
    while ( (i<tasks.size()) &&
                (task.getPriority() >
((Task)tasks.elementAt(i)).getPriority() )
              ) {
      i ++;
    }
tasks.insertElementAt(task, i);
}
SOLO ORDENA POR PRIORIDAD
```

Procesadores

El caso anterior solo modela por prioridad

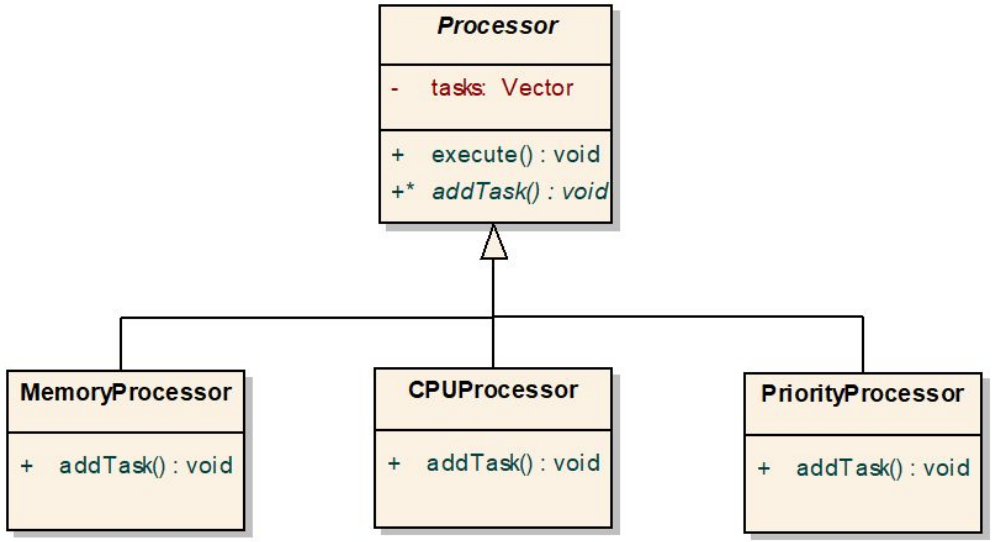
¿Cómo permito que se ordene por el uso de memoria y el uso de procesador ?

Procesador

Incorporo nuevos procesadores

add es **abstracto** en Processor

Entonces Processor es **abtracta**



Procesador: add en MemoryProcessor

– – –

```
public void addTask(Task tarea) {
      int i = 0;
      while ( (i < tasks.size()) &&
            ( tarea.getMemory()< ((Tarea)tasks.elementAt(i)).getMemory()  ) {
             i++;
      }
      if (i < tasks.size())
      tasks.insertElementAt(tarea, i);
      else
      tasks.add(tarea);
 }
```

Procesadores: add en CPUProcessor

– – –

```
public void addTask(Task tarea) {
        int i = 0;
        while ( (i < tasks.size()) &&
                ( tarea.getCPU()< ((Tarea)tasks.elementAt(i)).getCPU()  ) {
                 i++;
        }
        if (i < tasks.size())
        tasks.insertElementAt(tarea, i);
        else
        tasks.add(tarea);
 }
```

Procesadores: dudas

No son muy parecidos los dos códigos?

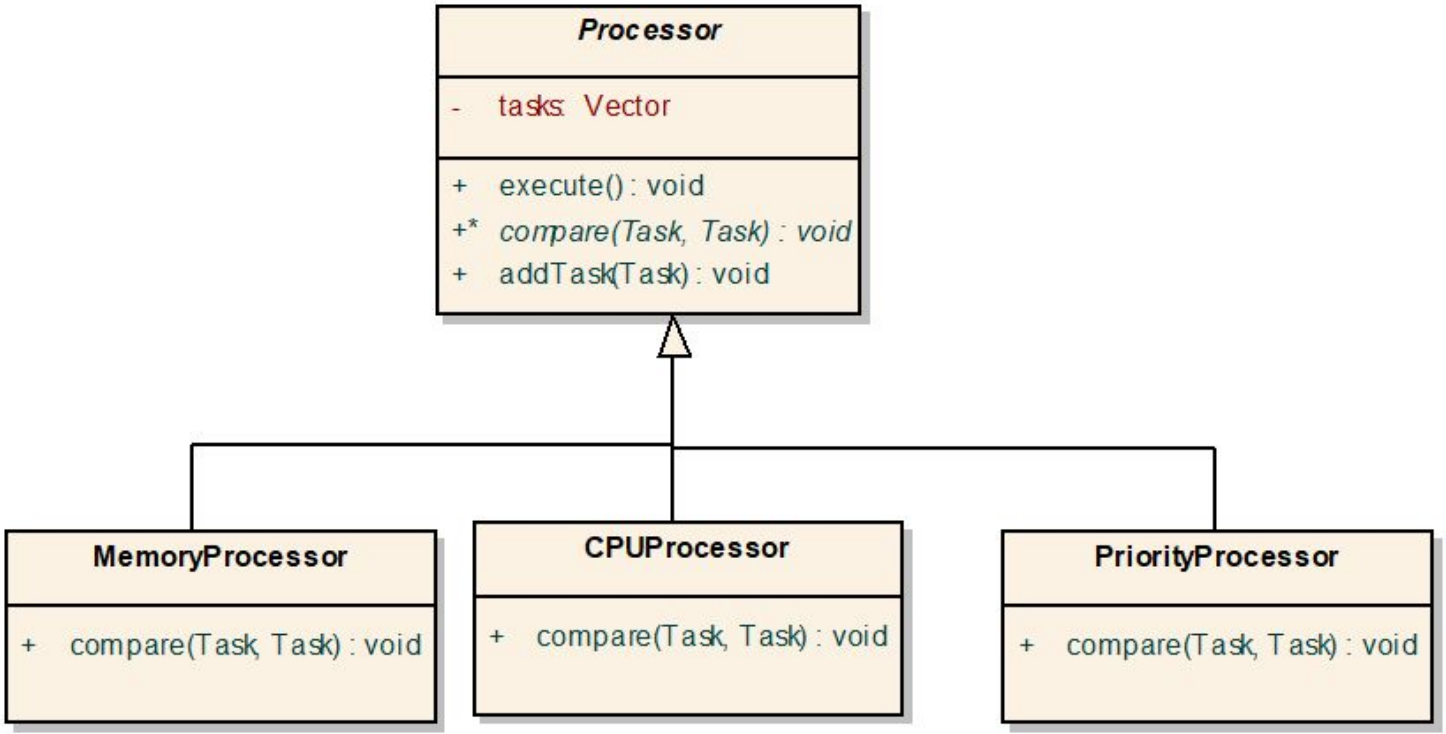
El procesador que ordena por prioridad, va a ser igual?

Procesadores: Abstracción del método add

En la clase Processor

```
public void addTask(Task tarea) {
        int i = 0;
        while ( (i < tasks.size()) &&  ( this.compare (tarea, ((Tarea)tasks.elementAt(i))  ) {
                i++;
        }
        if (i < tasks.size())
        tasks.insertElementAt(tarea, i);
        else
        tasks.add(tarea);
 }
public abstract boolean compare(Task  t1, Taskt2); // en Processor no se como implementarlo
```

Procesador: Clases



Procesadores: Clase CPUProcessor

Debe implementar el método abstracto compare

```
public boolean compare(Task t1, Task t2){
    return t1.getCPU()< t2.getCPU();
}
```

Solo provee la comparación con el CPU de la tarea

Procesadores

---

ahora se quiere incorporar un procesador que agrega las tareas por orden de llegada!!!!

La tarea nueva siempre va al final de la lista de tareas!

Procesadores

Hay que preguntarse:

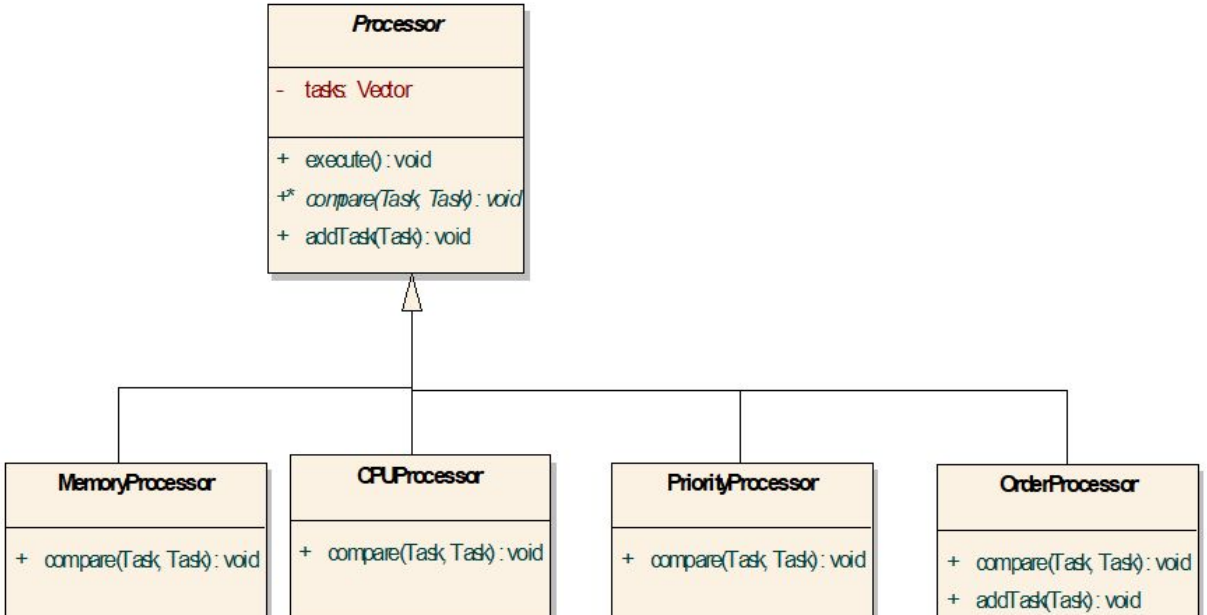
El procesador por orden de llegadas es un procesador?

SI

Tiene el mismo comportamiento que los demás?

SI, la misma interfaz (se le agregan tareas) pero cambia la forma en que las guarda

Procesadores



Procesadores: OrderProcessor

```
public class OrderProcessor extends Processor {

   public void add (Task tarea){  //Redefine comportamiento de la
clase Processor
       taks.add(tarea);
    }

   public boolean compare(Task t1, Task t2){
      return false;
   } // igual debe implementarlo porque es abstracto en
Processor
}
```

Si conocemos de antes que existe un procesador por orden de llegada la solución hubiese sido otra